(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 333 350 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
06.08.2003 Bulletin 2003/32

(51) Int Cl.[7]: **G06F 1/00**

(21) Application number: **02250644.8**

(22) Date of filing: **30.01.2002**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **STMicroelectronics Limited**
Almondsbury, Bristol BS32 4SQ (GB)

(72) Inventor: **Dellow, Andrew**
Strand, Gloucester GL5 3PX (GB)

(74) Representative: **Loveless, Ian Mark**
Reddie & Grose,
16 Theobalds Road
London WC1X 8PL (GB)

(54) Memory security device

(57) A semiconductor integrated circuit includes a processor for executing application code from a memory and a verifier processor arranged to receive the application code via the same internal bus as the processor. The verifier processor performs a verification function to check that the application code is authentic. The verifier processor runs autonomously and cannot be spoofed as it receives the application code via the same internal bus as the main processor.

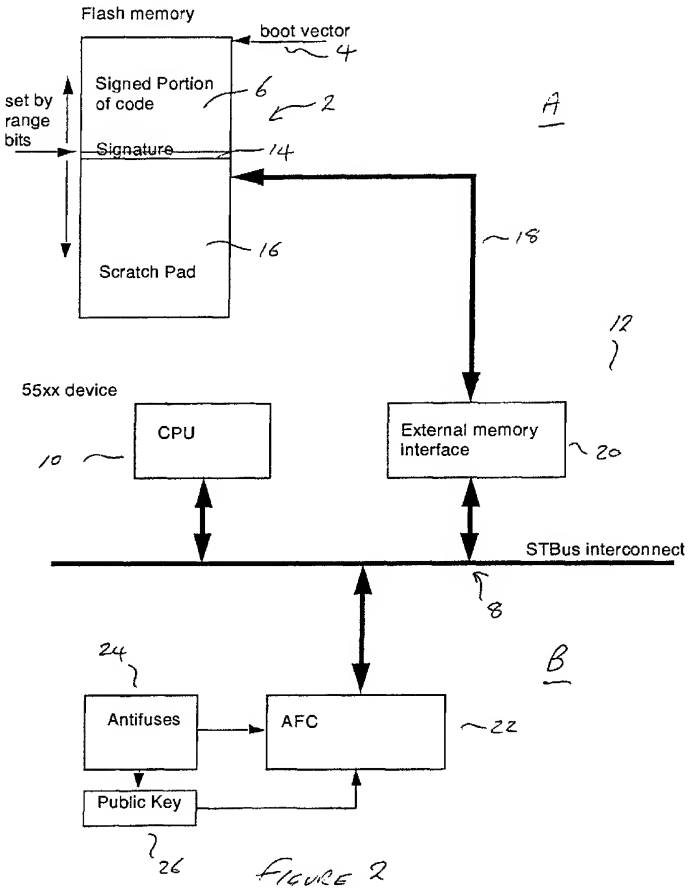

FIGURE 2

EP 1 333 350 A1

Description

FIELD OF THE INVENTION

[0001] The present invention relates to a memory security device, and in particular to the security of flash memory used in conditional access devices.

BACKGROUND OF THE INVENTION

[0002] In conditional access devices for pay television, or any other device using memory and requiring security, there is a need to provide flash memory but to avoid hacking. Hacking is the unauthorised placing of software in memory to override security features.

[0003] A known way of attempting to prevent hacking is to use some form of checking instructed by ROM memory to ensure that an application code stored in flash memory is correct. Such a device is shown in Figure 1.

[0004] A flash memory 2 has a boot sector 6 and an application sector 16. A CPU 10 is arranged to run application code from the flash memory 2 retrieved over an interface 12 via bus 8 EMI 20 and bus 18. The security is provided by the fact that CPU 10 boots from a boot ROM 3 which contains code to check the boot sector 6 of the flash memory. This is done once by the CPU producing a function of the code in the boot sector and comparing with a stored signature on startup. The CPU then jumps to the code in the boot sector 6 if it passes the check. However, we have appreciated that there is a relatively simple way of hacking such a security arrangement. When the CPU 10 boots up using code from the ROM 3, the CPU checks that the code in the boot sector 6 is correct. The weakness is that the process of power on, CPU boot and checking the flash takes a predictable number of clock cycles of the CPU clock. Thus to hack the system, a hacker places code in an unchecked part of the flash memory 2 and forces the CPU to read from that part of the memory after a predetermined number of clock cycles by fixing an external address line.

[0005] The CPU 10 thereafter runs from unchecked code and no further checks are conducted, because the verification of code is only conducted on boot up from the ROM 3.

[0006] We have appreciated the problem that memory storing application code within devices can be insecure and prone to hacking by storing unauthorised code.

SUMMARY OF THE INVENTION

[0007] The invention is defined in the independent claims with preferred features set out in dependent claims.

[0008] An embodiment of the invention comprises an additional processor termed a verifier processor and code arranged to read data from a memory to be checked, to produce a function of that data, and to verify that function of the data against a stored code. The verifier processor is on the same device and has the same external interfaces as a CPU which runs application code from the memory. The advantage of using an additional processor on the same device as a CPU is that the system cannot be hacked by changing code stored in memory as the additional processor would then also receive changed application code which would not be verified.

[0009] The verifier processor is arranged to continually check the flash memory whilst the CPU executes from the flash memory. If the address lines of the device were redirected so that the CPU runs from unauthorised code, then the verifier processor would also be redirected to that unauthorised code which would not pass the check.

[0010] The additional processor preferably produces a hash of the application code stored in memory and uses signature techniques to verify the application code.

[0011] The embodiment thus comprises an additional processor function which runs independently of a CPU but is within the same integrated circuit as that CPU and shares the same bus. The processor function analyses the application code applied to the CPU and, if not authentic, issues a reset signal to reset the integrated circuit.

BRIEF DESCRIPTION OF THE FIGURES

[0012] An embodiment of the invention will now be described by way of example only and with reference to the figures in which:

Figure 1: shows a known CPU memory arrangement for checking code;

Figure 2: shows an arrangement of a memory checking processor embodying the invention;

Figure 3: shows the memory checking processor of Figure 2 in greater detail.

DESCRIPTION OF AN EMBODIMENT

[0013] The Autonomous Flash Checker (AFC) comprises a processor whose purpose is to check or verify that application code stored in a flash memory 2 for execution by a CPU 10 is authentic and has not been changed or "hacked". Flash memory is used in many devices, but the preferred embodiment is a conditional access system for television broadcast. In such systems it is important that application software stored in flash memory is authentic and has not been changed in any way by a hacker. Such changes could be to run code whereby the system decrypts received broadcast signals without requiring payment by the user. The AFC processor 22 thus analyses the application code stored in flash memory to ensure the code is authentic.

[0014] An integrated circuit embodying the invention is shown in Figure 2, labelled device B. A processor

(CPU) 10 is connected via an internal bus 8 and an external interface 20 via external connections on an interface 20 via external connections on an interface 12 and bus 18 to a flash memory 2. The flash memory 2 contains application code in a boot sector 6 for execution by the CPU 10, signature portion 14 and an application sector 16. The signature portion contains a signature that is a function of the application code in the boot sector itself and is used for verification.

[0015] On power up of integrated circuit B, the CPU 10 is directed to the boot vector 4 of the flash memory 2 on a first integrated circuit, labelled circuit A. The application code is then retrieved over bus 18 and external memory interface 20 via internal bus 8 by the CPU 10 which executes the code. The application code is stored in a signed code portion 6 of the flash memory. It is noted that the AFC 22 is connected to the same internal bus 8 and external connections as the CPU, and so retrieves exactly the same code as the CPU without possibility of external interference. This is because the CPU, AFC processor and interconnect bus are all part of the same integrated circuit, labelled device B.

[0016] The CPU 10 and flash memory 2 operate in a known fashion, unless the AFC processor determines that the application code in flash memory 2 is not authentic, however, it impairs operation of the device B by using a reset causing the device to reset and the boot sequence is restarted. Thus, if the application code has been tampered with, the set top box will repeatedly reboot and will not function to decrypt received TV signals. Other forms of impairing the operation of device B could be used such as disabling or stopping the device clock or otherwise limiting the functionality of the device.

[0017] The operation of the AFC processor itself is by producing a hash function and a signature of the application code using a public key from a public key store 26 selected by antifuse 24 (Figure 2) as will now be described with reference to Figures 2 and 3.

[0018] The AFC 22 comprises a verifier processor such as a Risc processor 30 which executes code stored in code ROM 40 and uses RAM 12 for temporary storage. The code in code ROM 40 is only accessible by the verifier processor and instructs the Risc processor 30 to undertake the following steps:

1. Produce a hash of application code received from the flash memory.
2. Produce a signature function of the hashed code.
3. Verify that the signature is correct.

[0019] The verifier processor is not externally accessible other than in specific ways described later, and so cannot be hacked and only runs from the code in ROM which cannot be changed.

[0020] If the signature is correct then the application code in flash memory is deemed authentic.

[0021] The steps set out above are undertaken continually; each set of steps comprising a cycle of the verifier processor.

[0022] During each cycle the CPU 10 continues to operate as normal in retrieving and executing the application code over the same internal bus 8 as used by the verifier processor 30 over line 38. Accordingly, to avoid reducing the performance of the CPU 10, the verifier processor requests application code from the memory less frequently than the CPU, for example the verifier requests code once every 1,000 to 10,000 CPU requests. Also, the verifier requests are at pseudo random locations and at pseudo random times. This helps obscure the verifier requests amongst the CPU requests. The requests made at external connections at interface 12 for data from the flash thus comprise CPU requests and pseudo random requests at pseudo random times comparatively infrequently mixed together.

It is thus all but impossible for a hacker to determine how to spoof the external address lines to direct the CPU to hacked code but the verifier to genuine code. The use of pseudo random locations and times makes spoofing harder. The requests to the flash memory themselves are indistinguishable, whether made by the CPU or verifier processor.

[0023] The first step of producing a hash of the application code uses a flash read circuit which is instructed by RISC processor 30 to retrieve the code over bus 8. The application code can be read sequentially or in circular or pseudo random fashion as specified by the code in ROM, and is provided to the RISC processor 30 over line 38.

The hash function can be any one-way hash function which has the advantage that any small change in the application code will result in a large change in the hashed code, but is mathematically all but impossible to derive multiple changes that could be made to the application code such that the hashed code is unchanged. Preferably, the RISC processor 30 continually receives the application code from the flash in a pseudo-random read pattern, and uses a know hash function such as MD5.

[0024] On completion of the hash function, the second step is to produce a signature function of the hashed code. To do this, the RISC processor produces a function F (hashed code, public key) where the public key is selected from the public key store 26 by antifuse 24 and provided at 42. The signature of the application code is retrieved from signature location 14 (Figure 2) in the flash (having been created and prestored using the corresponding private key). A second function G (public key, signature) is then produced. The antifuses are arranged to select only one of a plurality of keys in the key store 26. This allows a generic device B to be created but to be tailored by selecting just one of possible keys. The antifuses are known and are irreversible fuses.

[0025] The third step is then to verify the hashed code against the signature by the standard digital signature technique of comparing F (hashed code, public key) and G (public key, signature). The preferred algorithm is

DSA.

Provided that the signature is verified, then no action is taken. If the application code does not verify the signature, however, an impair function results at 48 a chip reset is issued over line 50, preventing the chip operating further.

[0026] We have also appreciated that there may be a need to download new (authentic) code to the flash memory and that this should be provided for so that the AFC does not erroneously reject this new code. To allow this, the verifier processor must be stopped for M minutes, but again this could leave the possibility of a hack in which the AFC is permanently stopped. To prevent this, the code in ROM 40 causes the verifier processor to automatically issue a chip reset after M minutes, and starts the verification at the beginning of the new code in flash memory.

[0027] The only commands available to the CPU 10 to control the verifier processor 22 are: STOP, RESTART, PAUSE. Thereafter, the operation of the AFC processor is autonomous and largely in hardware, with the only software being in ROM 40 or RAM 42 which are only accessible by the RISC processor 30.

[0028] We have appreciated, however, that these commands need to be available to the CPU to avoid contention and allow flash memory updates, but could open the possibility of hacks which permanently pause or stop the AFC or continually reset. For that reason, further preferred features are included.

[0029] A first preferred feature is to allow the CPU 10 to pause the verifier processor to avoid contention. However, we have appreciated that this could allow a hack in which the AFC is permanently paused. So the code in ROM 40 is configured only to allow N pauses in one cycle of checking the flash memory. Each pause can be around 1 second. If the count N is exceeded, a chip reset signal is sent.

[0030] A further possible hack would be to block requests made by the verifier processor for code from the flash such that the verification process never completes. To prevent this, a watchdog function is incorporated in the code in ROM 40 such that a reset is issued if a cycle does not complete in a given time. The given time is predictable as it is known how long should be taken for a cycle and so this is programmed in ROM.

[0031] It could also be possible to hack the flash memory code such that the CPU continually instructs the verifier processor to stop and restart. To avoid this, the code in ROM does not allow the verifier processor to stop after a restart request so that a whole cycle of verification is conducted.

[0032] It is noted that the verifier processor has access to RAM 42. This RAM is also on the same device as the CPU, ROM 40 and verifier processsor so as to avoid the possibility of hacking this RAM which is used to store temporary values during execution of the verification code in ROM 40. The verifier processor only has two external connections; to retrieve data and to issue resets. The reset is issued to the device B itself and so cannot be hacked and the retrieval of code uses the same bus as the CPU and so if hacked would not be verified as previously described.

Claims

1. A semiconductor integrated circuit arranged to execute application code received from a memory via external connections, comprising: a processor for executing application code from the memory; an internal bus within the integrated circuit for providing the application code to the processor from the external connections; and **characterised by** a verifier processor arranged to receive the application code via the internal bus, wherein the verifier processor is arranged to process the application code using a verification function and to impair the function of the integrated circuit in the event that the application code does not satisfy the verification function.

2. A semiconductor integrated circuit according to claim 1, wherein the verification function includes a hash function on the application code.

3. A semiconductor integrated circuit according to claim 1 or 2, wherein the verifier processor is arranged to receive a stored secret from the memory and the verification function is a comparison of the secret and the processed application code.

4. A semiconductor integrated circuit according to any preceding claim, wherein the verification function comprises hashing the application code to produce hashed code, retrieving a signature of the code from a signature store within the memory and verifying the hashed code and signature using a public key.

5. A semiconductor according to any preceding claim, wherein the verifier processor has a pause input arranged to receive a pause signal from the processor, and is arranged to impair the function of the integrated circuit if greater than a given number of pause signals are received in a given time.

6. A semiconductor integrated circuit according to any preceding claim, wherein the verifier processor has a stop input arranged to receive a stop signal from the processor, and is arranged to issue impair the function of the integrated circuit a given time period after receiving a stop signal.

7. A semiconductor integrated circuit according to any preceding claim, wherein the verifier processor has a stop input and is arranged to restart a given time period after a stop, and arranged not to stop again until completing the verification function on the code

at least once.

8. A semiconductor integrated circuit according to any preceding claim, wherein the verifier processor is arranged to impair the function of the integrated circuit if the verification function is not completed within a predetermined time.

9. A semiconductor integrated circuit according to any preceding claim, wherein the verifier processor requests portions application code from the flash memory at intervals between requests by the processor for portions of the application code.

10. A semiconductor integrated circuit according to claim 9, wherein the verifier processor requests portions of application code less frequent intervals than the processor.

11. A semiconductor integrated circuit according to claim 9 or 10, wherein the verifier processor is arranged to request portions of application code at pseudo random times.

12. A semiconductor integrated circuit according to claim 9, 10 or 11, wherein the verifier processor is arranged to request portions of application code from pseudo random locations in the memory.

13. A semiconductor integrated circuit according to any preceding claim, comprising a ROM arranged on the same integrated circuit, wherein the verification function is defined by the code in the ROM.

14. A semiconductor integrated circuit according to claim 13, wherein the ROM is only accessible by the verifier processor.

15. A semiconductor integrated circuit according to claim 13 or 14, further comprising a RAM on the same integrated circuit for storage of temporary values when executing the verification function.

16. A semiconductor integrated circuit according to any preceding claim, wherein impairing the function of the integrated circuit comprises resetting the circuit.

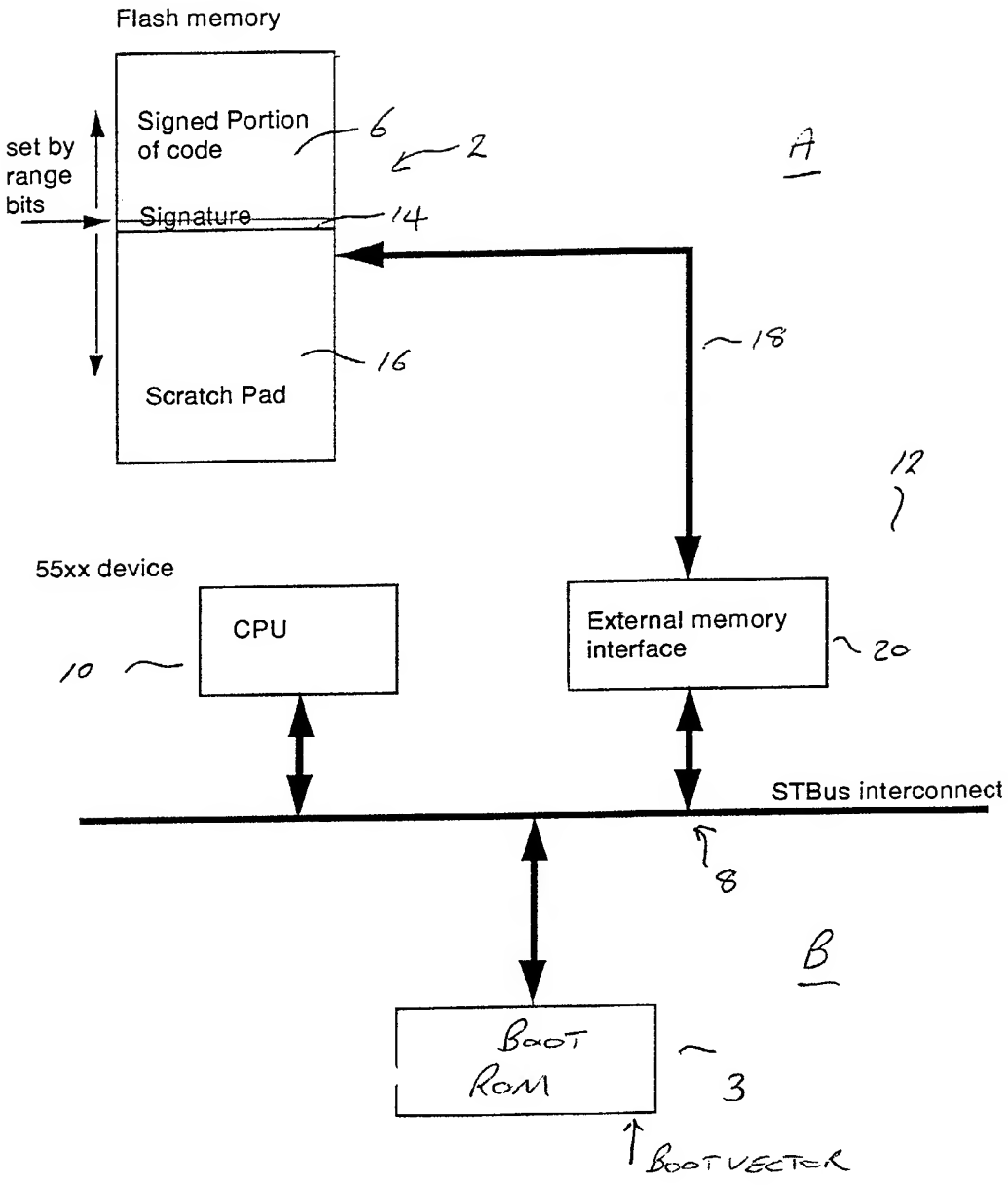
Flash memory
Signed Portion of code
6
2
A
set by range bits
Signature
14
18
16
Scratch Pad
12
55xx device
CPU
10
External memory interface
20
STBus interconnect
8
B
Boot ROM
3
Boot vector

FIGURE 1

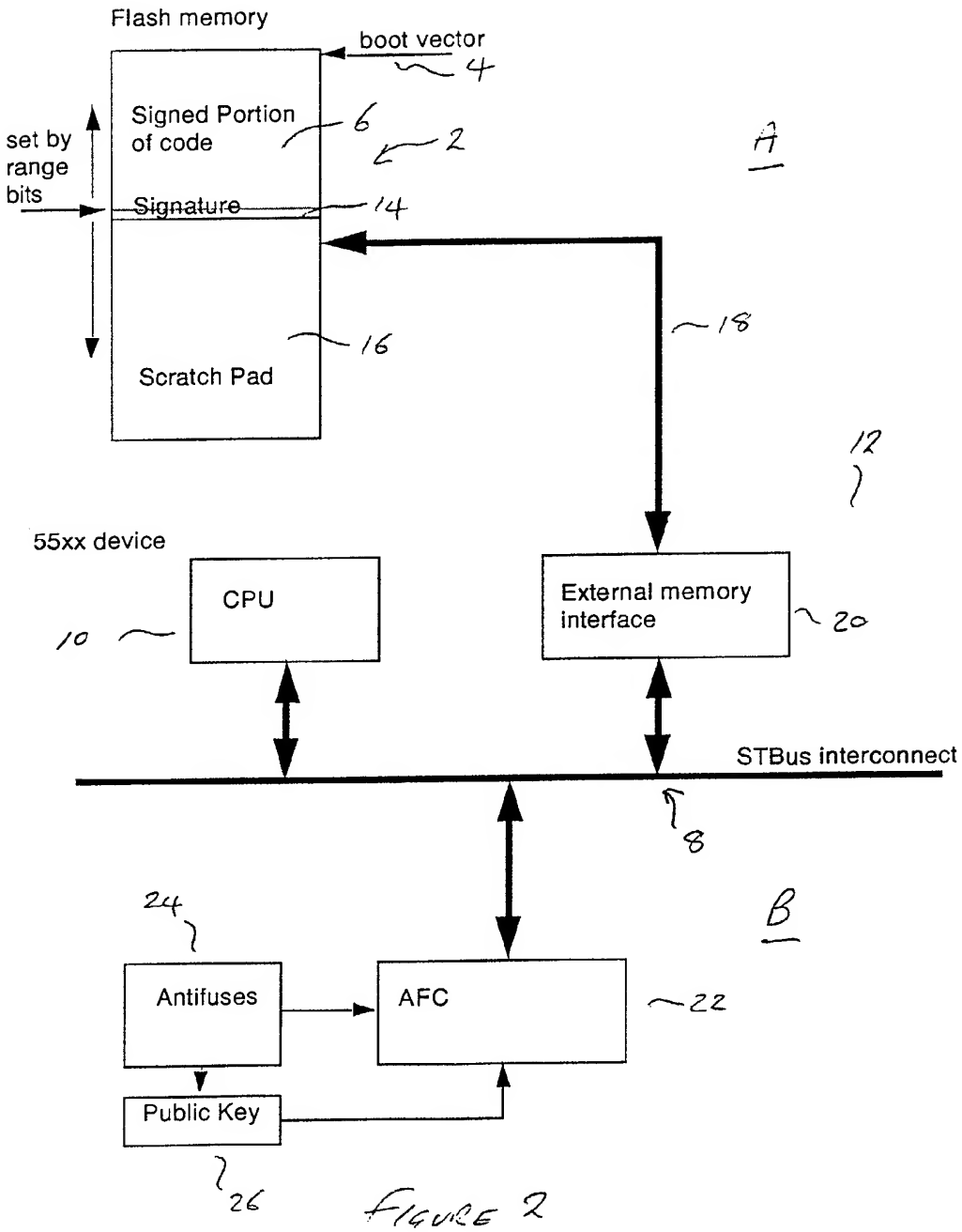
Flash memory
boot vector
4
Signed Portion of code
6
2
A
set by range bits
Signature
14
16
18
Scratch Pad
12
55xx device
CPU
10
External memory interface
20
STBus interconnect
8
B
24
Antifuses
AFC
22
Public Key
26

FIGURE 2

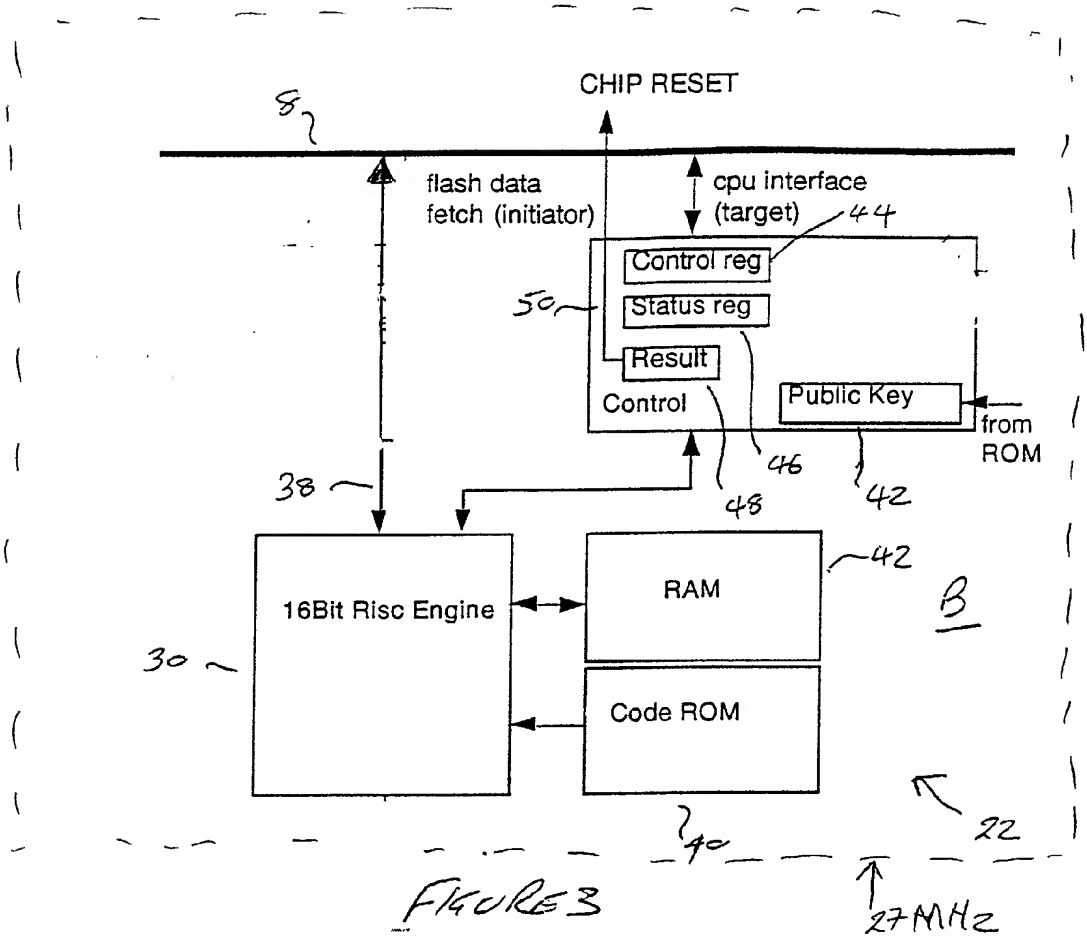
CHIP RESET
8
flash data
fetch (initiator)
cpu interface
(target)
44
Control reg
Status reg
Result
50
Control
Public Key
from
ROM
46
48
42
38
16Bit Risc Engine
RAM
42
B
30
Code ROM
40
22
27MHz

FIGURE 3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 25 0644

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X Y	US 6 311 273 B1 (ACKERMAN III WILLIAM H ET AL) 30 October 2001 (2001-10-30) * column 1, line 55 - column 2, line 4 * * column 2, line 55 - column 3, line 44 * * column 4, line 9 - line 37 * * column 6, line 63 - column 7, line 8 * * column 7, line 35 - column 8, line 14 * * column 8, line 44 - column 9, line 23 * * figure 1 *	1,2,4, 13-16 3,8	G06F1/00
Y	EP 0 962 850 A (NOKIA MOBILE PHONES LTD) 8 December 1999 (1999-12-08) * column 2, paragraph 6 * * column 3, paragraph 11 - column 4, paragraph 13 * * figure 3 *	3	
Y	US 6 009 523 A (NAGASHIMA TAKESHI ET AL) 28 December 1999 (1999-12-28) * column 6, line 52 - column 7, line 18 *	8	
A	"ARTIFICIAL IMMUNITY FOR PERSONAL COMPUTERS" IBM TECHNICAL DISCLOSURE BULLETIN, IBM CORP. NEW YORK, US, vol. 34, no. 2, 1 July 1991 (1991-07-01), pages 150-154, XP000211065 ISSN: 0018-8689 * page 152, paragraph 1 * * page 152, last paragraph *	9,10	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	17 June 2002	Arbutina, L

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 25 0644

This annex lists the patent family membersrelating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

17-06-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6311273 B1	30-10-2001	US 6038667 A	14-03-2000
		US 5953502 A	14-09-1999
		EP 1013023 A1	28-06-2000
		JP 2001524229 T	27-11-2001
		WO 9836517 A1	20-08-1998
EP 0962850 A	08-12-1999	FI 981232 A	02-12-1999
		EP 0962850 A2	08-12-1999
US 6009523 A	28-12-1999	AU 686494 B2	05-02-1998
		AU 4632496 A	27-08-1996
		BR 9605115 A	07-10-1997
		CA 2187038 A1	15-08-1996
		CN 1146814 A	02-04-1997
		EP 0754999 A1	22-01-1997
		WO 9624894 A1	15-08-1996
		KR 268693 B1	16-10-2000
		ZA 9601013 A	29-08-1996

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82